International Journal of Emerging Trends in Computer Science and Information Technology
ISSN: 3050-9246 | https://doi.org/10.63282/3050-9246.IJETCSIT-V4I2P126
Eureka Vision Publication | Volume 4, Issue 2, 263-284, 2023

Original Article

A Scalable Data Governance Architecture for Establishing Reliable Data Lineage, Metadata Management, and Quality Controls in Hybrid Enterprise Data Platforms

Sudha Donepudi
Independent Researcher, USA.

Abstract - Hybrid enterprise data platforms share a single business fact across on-premises databases, enterprise data warehouses, data lakes, cloud storage, distributed processing engines and a range of streaming channels. Governance designed for one warehouse does not scale: metadata is fragmented; lineage exists only where instrumentation happens to exist, quality controls attach to pipelines rather than to the assets they protect, and policy enforcement differs by platform. In this article, scalable data governance architecture is proposed, which incorporates metadata standardization, end-to-end lineage and provenance tracking, embedded quality controls and consistent policy enforcement across heterogeneous platforms. The architecture stems from a well-structured analysis of governance frameworks, metadata and provenance studies, quality standards and documented enterprise systems and carves out a separation of a central governance control plane from distributed data ownership. No implementation was developed, and no dataset or measurement instrument was available; therefore, no experimental performance results are reported. The article contributes a twelve-component architecture traceable to functional and nonfunctional requirements, defined behavior under six operational stress cases, ten reproducible metric definitions, and an evidence-based analysis separating findings reported by prior sources from untested proposals. Empirical validation remains future work.

Keywords - Data Governance, Data Lineage, Metadata Management, Data Quality, Hybrid Data Platforms, Data Provenance.

1. Introduction

Enterprise data platforms increasingly span on-premises systems, cloud storage, distributed processing engines, and streaming services, creating new challenges for consistent data governance. Systems of record stayed on premise, analytical workloads started to partially migrate to cloud databases and object storage, transformation got rid of monolithic extract/transform/load jobs and started to be orchestrated through workflows executed by distributed engines, and streams began to feed analytical stores independently of nightly windows. Each platform carries its own schema model, access-control semantics and audit surface, so the question is not whether data can be governed but whether governance stays coherent, traceable and affordable once data, processing and consumers span systems never designed to be governed together.

The reported failure modes do not vary. Usually, governance mechanisms (decision rights, policies, standards and monitoring) are studied separately, rather than as an aligned configuration [3] [1] and the optimal configuration is dependent on the contingency factors of regulatory exposure and organizational structure [2]. Quality research reaches a parallel conclusion: of thirteen assessment and improvement methodologies compared, many adopt only a data-driven or only a process-driven strategy, [20]. Standards were already well specified, through the SQuaRE quality model [21], the ISO 8000 management process reference [22] and the domain-agnostic provenance model [10], which locates the gap in architecture rather than in definition.

Capabilities that have to work together are designed independently or within a single platform boundary, such as lineage, metadata, quality and policy. A key research gap is the absence of a requirement-mapped architecture for hybrid platforms that specifies how lineage, metadata, and quality controls interact, how the architecture behaves when pipelines fail or schemas evolve, and how its performance should be evaluated.

Five research questions guide the study, as described in Section III-B. The study adopts an evidence-based approach to architectural analysis, linking each design decision to documented findings or an explicitly stated design rationale. The literature review focuses on relevant governance, metadata, lineage, provenance, and data quality research, including foundational studies and later platform documentation where needed to explain the proposed architecture. No implementation was developed, and no empirical measurements were collected. The contributions are: C1, a twelve-component, four-layer architecture specifying responsibilities, interfaces, and flows; C2, a coupling model linking metadata records, lineage edges, quality rules, and policy classifications to a canonical asset identity; C3, defined behaviour under six operational stress cases;

C4, ten reproducible metric definitions accompanied by a requirement-to-control traceability matrix; and C5, an evidence analysis distinguishing findings reported by prior sources from untested architectural proposals.

2. Background and Related Work

2.1. Governance Principles, Roles and Accountability

Data governance concerns the decision rights and accountability framework determining how data is created, managed and consumed. A structured review breaks the field into six dimensions, and identifies governance mechanisms as an aligned configuration, not as a checklist [3]; earlier research identified the design space based on who exercises decision rights and how [1]; and contingency research indicated that the amount of centralization needed depends on the organizational context and regulatory setting, suggesting no single best design [2]. In practice bodies, the same territory is divided into knowledge areas that have specific owner and steward roles [4].

2.2. Metadata Management, Repositories and Catalogues

Technical metadata describes the structure, business metadata describes the meaning, and operational metadata describes what is done at run time; the last of these is the only one that scales automatically, while the first is harvested and the second is curated by humans, so these are three different categories of metadata, an architectural distinction [4]. The designs of the catalogue reflect that asymmetry: one is a production system that collects six metadata groups and derives relationships from the data itself, given that user-supplied descriptions are not always available [12]; the other is an argument that disaggregated stacks have no common context service that captures the applications, behaviour, and change [13]. An open governance platform identifies extensible types, entities and classifications, and cascades classifications through lineage instead of storing them per platform [15]. There are proposals for decentralised architecture [23] that argue for a single repository or some kind of federation of domain stores that share common standards; the interface is a catalogue, the system of record is the repository itself that it reads.

2.3. Lineage and Provenance

Lineage research starts with the origin of derived data: in a warehousing environment, the lineage of view data is derived from a set of contributing tuples, attributes and source relations [5]. In provenance theory, why-provenance is defined as specifying which combination of source tuples is sufficient for an output tuple and how-provenance specifies the operations applied to the tuples; in [6] this separation of why and where provenance was implemented; a subsequent synthesis [7] grouped these questions into why, how and where; surveys revealed a variety of models and degrees of granularity capturing provenances for different purposes, limiting reuse across systems [8] [9]. The PROV data model defines provenance as a record of the entities, activities and agents involved in producing, influencing or delivering data, with relations, and is domain-agnostic and extensible [10]. The two differences in Section IV are lineage as a navigable dependency graph over assets important for impact analysis and defect triage versus provenance as a record of which activity and agent produced a specific version important for audit and that lineage can be declared from pipeline definitions or derived from execution, declarative capture being complete for governed pipelines, but blind to all dynamic statements, and execution capture observing what ran at the cost of instrumentation, and gaps when a job fails.

2.4. Transformation Tracking In Etl and Elt Pipelines

Pipeline metadata adds lineage to the pipeline. The warehouse layer has introduced an explicit metastore that is separate from the data and explicitly mandates consistency between metadata and data [11]; orchestration frameworks provide the workflow graph with dependencies and run outcomes, formalized as run-level models, that treat jobs, datasets and runs as separate instrumented entities [16]; storage layers can provide transaction logs recording metadata changes and commit provenance, such as the user who committed the changes [17]. Many of the lineage sources available today already contain a lot of what the programme needs, but these come from different systems, use different models and refer to assets differently, so integration is the only limiting factor, hence the need to stitch incomplete traces explicitly, sacrificing precision for recall in production lineage systems [24].

2.5. Data Quality Dimensions, Measurement and Remediation

The quality research starts with fitness for use, which has already been established by the data consumers empirically from their values on the product and not from its intrinsic correctness [19] and comparison methods converge to accuracy, completeness, consistency and timeliness with many more dimensions having different definitions [20]. The SQuaRE model standardised the vocabulary, dividing attributes by their inherent and system-dependent perspectives into fifteen characteristics and asserting that their significance is unique to different stakeholders [21], while the ISO 8000 series provides a reference for process capability evaluation through data quality management processes [22]. Two distinctions matter. Measurement and remediation are separate activities; measurement is a statistical measurement across a population of a given attribute at one point in time, dependent on sampling and the denominator selected, while remediation is the modification of the data or the process which generated it. And improvement is frequently underspecified with thin evidence on its economics [20], so neither the cost nor the effect of a control may be assumed.

2.6. Hybrid Platforms, Monitoring and Open Problems

The hybrid estate is not a transitional estate. Distributed processing introduced the separation of storage and computation, and schema on read was acceptable [11] cloud object stores further reduced storage costs at the expense of absorbed latency [17] and streaming turned some inputs into continuous data. The decentralized-architecture argument is centered on domain-oriented ownership, data as a product, self-serve infrastructure, federated computational governance, and the capability of central catalogue and lineage being placed in the self-serving computational platform because the problem of discovering and knowing the provenance of data cannot be solved by its domains individually [23]. Typically classification becomes reporting, but often it is also classification of assets as sensitive or regulated leading to authorization and masking, and classification of assets as sensitive or regulated being propagated as lineage, so the label is carried along with the data through processing [15].

Four conclusions follow from the reviewed evidence. First, effective data governance depends on the alignment of governance mechanisms with organisational and regulatory context [2], [3]. Second, metadata fragmentation cannot be resolved simply by collecting more metadata; heterogeneous descriptions require standardisation, contextual mapping, and, where appropriate, inference [12], [13]. Third, lineage completeness must be explicitly assessed, and unresolved relationships must be represented rather than silently omitted [24]. Fourth, although data quality assessment methods and standards are established, the effectiveness and economic implications of improvement processes remain insufficiently evidenced [20], [22]. The architectural gap is the lack of a unified, requirement-mapped approach that connects these concerns across platforms and defines how its behaviour should be evaluated under operational stress.

Table 1. Comparative Analysis of Existing Data Governance Research and Documented Approaches

Ref.	Year	Research Objective	Architecture / Framework	Metadata Approach	Lineage Capability	Quality Controls	Findings As Reported	Limitations Identified
[1]	2010	Define the decision-rights design space for data governance	Governance design framework built on IT-governance matrices	Not specified beyond the need to assign ownership of definitions	Not addressed	Not addressed	Governance design requires explicit allocation of decision rights across organisational units	Conceptual; no architecture, no implementation, no metrics
[2]	2009	Test whether data governance design is contingent on context	Contingency framework of governance forms and mechanisms	Governance of definitions and ownership treated as a mechanism	Not addressed	Not addressed	No single governance design fits all organisations; design must follow contingency factors	Small comparative base; no technical components
[3]	2019	Synthesise data governance research into one conceptual framework	Six-dimension conceptual framework	Metadata and definitions located within governance mechanisms	Not addressed as a technical capability	Not addressed	Governance mechanisms are best understood as an aligned configuration, not a checklist	Conceptual review; no implementation guidance or evaluation
[4]	2017	Provide a comprehensive body of knowledge for data management	Knowledge areas covering governance, metadata, quality, architecture, security	Splits technical, business and operational metadata; defines roles	Lineage acknowledged as part of metadata management	Data quality positioned as a management area with dimensions	Professional practice reference synthesising practitioner consensus	Prescriptive rather than empirical; no measured outcomes
[5]	2000	Trace which source data contributes to derived view	Lineage tracing algorithms for view data	Warehouse metadata used as the input to	Tuple- and attribute-level lineage for relational	Not addressed	Lineage can be computed from query structure and	Relational scope; no heterogeneous platforms, no

		data in a warehouse		tracing	views		materialisation relationships	run-time instrumentation
[6]	2001	Characterise provenance questions formally	Why-provenance and where-provenance formalisation	Not applicable; provenance derived from data and query	Contributing-tuple and source-location semantics	Not addressed	Distinguishes why an output exists from where values came from; both are needed	Theoretical; no enterprise platform treatment
[7]	2009	Survey provenance in databases	Survey of why, how and where provenance	Provenance metadata as stored annotations	Lineage at tuple/attribute granularity	Traceability dimension noted in quality literature	Provenance storage and propagation trade-offs are explicit design decisions	Database-centric; limited treatment of pipelines and streaming
[8]	2017	Survey provenance purposes, forms and sources	Three-question survey framework	Provenance records as metadata objects	Captures work-, data- and workflow-centric provenance	Not addressed	Provenance models and granularities are heterogeneous, limiting cross-system reuse	Survey only; no architecture, no scalability evidence
[9]	2018	Systematic review of provenance systems	Comparative review of provenance systems and their features	Provenance store and model compared across systems	Capture, storage, query and visualisation compared	Not addressed	Provenance systems differ widely in model, capture mechanism and query support	Accessible record provided title, venue and abstract only; feature detail not extractable
[10]	2013	Standardise the interchange of provenance information	W3C PROV data model: entity, activity, agent plus seven core relations	Provenance records are structured metadata with a normative vocabulary	Derivation relationships expressible but not optimised for traversal	Not addressed	A domain-agnostic, extensible provenance model exists and is normative	Specification only; no operational, latency or retention guidance
[11]	2009	Provide warehouse-style data management over a distributed file system	Metastore plus query compiler and execution engine	Explicit metastore holds schemas, partitions and serialisation formats	Table and partition level; no transformation lineage	Not addressed	In production the warehouse held >700 TB in several thousand tables with >5000 queries/day	Authors state the platform must explicitly maintain consistency between metadata and data
[12]	2016	Organise structured datasets at enterprise scale	Post-hoc, non-invasive catalogue and search system	Six metadata groups; schema, provenance, ownership, usage, temporal	Upstream and downstream datasets recorded as provenance metadata	Not addressed	Catalogue indexed >26 billion datasets; >1.6 billion paths added/deleted per day; >20 formats	Metadata supply is uneven; ranking, missing metadata and semantics remain open problems
[13]	2017	Provide a shared data	Common Ground	Applications, behaviour	Lineage captured as	Not addressed	Absence of shared	Version 0 prototype;

		context service for disaggregated stacks	metamodel with model graph, version graph and lineage layer	and change as the three metadata dimensions	relationships between versions via workflows or principals		metadata services is described as a crisis point for large data estates	authors report no large-scale production deployment
[14]	2013	Curate heterogeneous sources at scale	End-to-end curation system with schema integration and entity consolidation	Global schema with attribute identification and mappings	Not addressed	Entity consolidation performs record-level cleaning	~90% lower curation cost; 90% correct attribute mapping after <50 labelled sources	No live feeds; no inter-entity relationship discovery; order-sensitive integration
[15]	2019	Provide open metadata management and governance for Hadoop and beyond	Type system with entities, classifications and lineage propagation	Predefined and user-defined types with inheritance; REST interfaces	Process entities with inputs and outputs capture dataset lineage; UI and REST access	Data-quality classification type defined; quality itself not evaluated	Classification propagates along lineage and can drive authorisation and masking	Platform-family scope; no cross-platform benchmark published in the documentation
[16]	2019 –	Collect, aggregate and visualise lineage metadata across a data ecosystem	Metadata service with HTTP API, UI and OpenLineage event model	Dataset, job and run entities with facets	Run-level provenance; dependencies between jobs and datasets; lineage graph	Not integrated; quality metadata external	Documents a generic run-job-dataset model with extensible facets for instrumentation	Documentation is later than 2019 and describes capability, not measured behaviour
[17]	2020	Provide ACID table storage over cloud object stores	Transaction log over Parquet objects	Log records metadata changes, file add/remove with statistics, protocol version	Commit provenance recorded per operation; table lineage with input files	Not addressed	1M-partition metadata query in 108 s (17 s cached); 93% data skipping on a 500 TB table	Single-table serialisable transactions; object-store latency bounds streaming
[18]	2016 – 2018	Provide a data quality service for batch and streaming data	Define, measure and report cycle over Hadoop and Spark	Quality rules as a domain model and DSL	Not addressed	Accuracy, completeness, timeliness and profiling among the covered dimensions	Unified measurement process across batch and streaming with an extensible rule language	Project documentation; project has since been retired
[19]	1996	Establish what data quality means to data consumers	Empirical two-stage survey and quality category framework	Not applicable	Traceability appears among the dimensions consumers value	Defines quality as fitness for use across multiple dimensions	Quality is multi-dimensional and use-dependent, not intrinsic accuracy	Survey of consumers; no measurement or remediation method
[20]	2009	Compare data quality assessment and improvement methodologie	Comparative survey of 13 methodologies across six perspectives	Not addressed	Not addressed	Dimensions, strategies, phases and costs compared	Many methodologies are single-strategy; several omit execution of	Empirical evidence on the economics of data quality is scarce

		s					improvement steps	
[21]	2008	Standardise a general data quality model for structured data	Fifteen quality characteristics in inherent and system-dependent viewpoints	Not addressed	Not addressed	Defines quality attributes usable for requirements, measures and evaluation	Provides a shared vocabulary; importance differs between stakeholders	Standard scope text only was accessible; no operational guidance in the accessible portion
[22]	2016	Standardise data quality management processes	Process reference model for data quality management	Not addressed	Not addressed	Processes for enhancing quality and assessing capability or maturity	Asserts process structure as the basis for quality management and maturity assessment	Process list not readable without purchase; scope statement only
[23]	2019	Propose an alternative to the monolithic data lake	Distributed domain-oriented architecture with federated governance	Data product registration with a central catalogue for discoverability	Lineage provided as a self-serve platform capability	Data quality framed as an attribute of the data product	Four principles: decentralised ownership, data as a product, self-serve platform, federated governance	Practitioner article; no implementation evidence or metrics
[24]	2019	Track data lineage for big data systems	Provenance framework combining database-style and workflow-style provenance	Lineage metadata maintained separately from the data	Unifies database and workflow provenance for big data jobs	Not addressed	Design merges two provenance traditions to expose lineage for distributed processing	Abstract-level access only; no architecture detail or performance figures verified

3. Research Problem, Objectives and Requirements

3.1. Statement of the Problem

Coupling and continuity are the two governance issues. The platform storing the asset, the orchestrator moving the asset, the catalogue indexing the asset, and the glossary defining the asset all hold metadata about the asset, and there's no guarantee that such metadata continues to point to the same asset after an asset has been renamed or moved or after a partition has been re-partitioned. Lineage is recorded in the presence of at least one instrument, and coverage is therefore uneven, with the gaps being indistinguishable from absence of relationships. Quality is measured and evaluated within pipelines which cannot be tied to a governed asset or to the governed asset's rule version. Each platform has its own respective policy and a classified attribute could be hidden in one environment, and visible in another. The goal is an architecture that retains asset identity, metadata, lineage, asset quality results and policy decisions throughout the changing of the estate.

3.2. Research Questions

The objective is operationalized by five questions. RQ1: How to capture and sustain lineage across heterogeneous source, transformations and consumption points, and how to represent incompleteness? RQ2: How to ensure that technical, business and operational metadata are standardized and synced between platforms without one platform enforcing the model? RQ3: What about embedding quality controls as part of ingestion, transformation, storage and consumption so that results can be traced to a rule version and an asset version? RQ4: What are the policies and audit evidence enforcement and documentation requirements to ensure consistent access-control semantics across platforms? RQ5: What are the ways to scale the architecture as assets, pipelines, users and workload grow and what would be proof of scaling?

3.3. Functional and Nonfunctional Requirements

Requirements were identified in two manners: literature-derived requirements are a result of a documented mechanism or stated limitation in the literature reviewed, and author-proposed requirements are a result of the coupling objective above.

Table II states fourteen functional and eight nonfunctional requirements with the responsible component, expected behaviour, the validation method that would test them and their origin. The functional set includes lineage acquisition for three different channel types, canonical asset identity, metadata harvesting and business-metadata binding, versioned quality rules that are evaluated at four points, remediation routing, policy evaluation with classification propagation, append-only audit and governed interfaces. The nonfunctional set includes: domain-partitioned scalability, degradation instead of blocking, stated bounds on freshness, bounds on divergence, reconstructibility, bounded metadata access, observability, extensibility. No requirement in Table II has been implemented or validated.

Table 2. Functional and Nonfunctional Requirements for the Proposed Architecture

Id	Category	Requirement Description	Responsible Component	Expected Behaviour	Validation Method	Supporting Reference	Origin
FR-01	Functional	Capture lineage from declarative pipeline and transformation definitions	ETL/ELT orchestration; metadata collection	Planned read-write relationships emitted before execution	Compare declared edges against a manually reviewed pipeline sample	[16]	Literature-derived
FR-02	Functional	Capture lineage from execution events for every instrumented run	Lineage and provenance engine	Job, input, output and parameter recorded per run	Reconcile captured events with orchestrator run history	[16], [24]	Literature-derived
FR-03	Functional	Infer lineage from executed statements and query logs where not declared	Lineage and provenance engine	Candidate edges emitted with reduced confidence and origin marker	Spot-check inferred edges against reproduced transformations	[12], [24]	Literature-derived
FR-04	Functional	Represent asset-level and column-level relationships with confidence and gaps	Lineage and provenance engine	Every edge carries origin and confidence; unresolved edges stored as explicit gaps	Structural validation of the graph; gap inventory review	[24], [7]	Author-proposed
FR-05	Functional	Issue and resolve a canonical asset identifier mapped to platform names	Metadata repository and catalog	Identity survives rename, migration and multi-platform presence	Rename and migration simulation; identity collision audit	[13], [12]	Author-proposed
FR-06	Functional	Harvest technical and operational metadata from each platform programmatically	Metadata collection and standardisation	Scheduled harvest with recorded mapping to the canonical model	Re-harvest comparison against platform catalogues	[11], [12]	Literature-derived
FR-07	Functional	Bind curated business metadata to physical assets through a governed glossary	Glossary and stewardship services	Terms mapped to assets; competing definitions retained and flagged	Glossary coverage audit; conflict register review	[4], [12]	Literature-derived
FR-08	Functional	Define, version and approve quality rules with effective dating	Data quality service	Every evaluation records the rule version applied	Rule version traceability test on historical evaluations	[22], [18]	Literature-derived
FR-	Functional	Evaluate quality	Data quality	Outcome	Controlled fault	[20], [19]	Literature

09		rules at ingestion, transformation, publication and consumption	service	recorded per evaluation point with pass, fail or not-evaluable status	injection at each evaluation point		-derived
FR-10	Functional	Route failures to quarantine and to a named accountable owner	Quality service; glossary and stewardship services	Quarantined data retains context; owner notified with severity and deadline	Exception lifecycle audit; notification acknowledgement rate	[20]	Author-proposed
FR-11	Functional	Evaluate classification, access, masking, retention and residency decisions	Governance policy and control enforcement	Decision returned with policy version and attached obligations	Policy decision comparison against manual review	[15]	Literature-derived
FR-12	Functional	Propagate classification along lineage to derived assets	Policy enforcement; lineage engine	Derived asset inherits the strictest classification of its inputs	Propagation test across multi-hop derivation chains	[15]	Literature-derived
FR-13	Functional	Persist append-only audit evidence for every governance decision	Monitoring, audit logging and reporting	Asset version, rule version, policy version, actor and outcome recorded; no update or delete	Attempted mutation test; evidence completeness sampling	[10]	Author-proposed
FR-14	Functional	Expose governance capability through component interfaces and a standards-based export	API and integration interfaces	Registration, traversal, evaluation and query operations available asynchronously	Interface conformance and interoperability test	[10], [15]	Literature-derived
NFR-01	Nonfunctional	Scale metadata storage and lineage traversal horizontally by domain	Repository; lineage engine	Growth in one domain does not degrade another; partitioning configurable	Load test with independent scaling of asset count and concurrency	[12]	Author-proposed
NFR-02	Nonfunctional	Governance services degrade rather than block data production	All governance services	Ingestion continues when a governance service is unavailable; events queued and reconciled	Availability fault injection with production continuity check	[13]	Author-proposed
NFR-03	Nonfunctional	Ensure metadata consistency within a declared maximum staleness bound	Metadata collection; repository	Divergence detected by reconciliation and reported with both versions	Latency measurement per channel; divergence detection test	[11]	Literature-derived
NFR	Nonfunctiona	Keep every	Audit store;	Historical	Reconstruction	[10], [7]	Literature

-04	l	governance assertion reconstructible from persisted evidence	repository	decisions reproducible after assets or rules change	test on a sampled historical decision		-derived
NFR -05	Nonfunctiona l	Secure governance metadata at least as strictly as the data it describes	Security, access control and compliance	Asset inventory, classifications and usage restricted by entitlement	Access-control review; entitlement testing	[4], [15]	Author-proposed
NFR -06	Nonfunctiona l	Report governance health, coverage and latency to a common monitoring surface	Monitoring and reporting; all services	Per-service health and per-requirement coverage visible and alertable	Observability audit; alert latency measurement	[18]	Author-proposed
NFR -07	Nonfunctiona l	Allow new platforms and rule types to be added by configuration	Metadata standardisation ; quality service	New connector or rule type requires no change to consuming services	Extension test with a new connector and a new rule type	[15]	Author-proposed
NFR -08	Nonfunctiona l	Keep data quality validation overhead within a declared operational budget	Data quality service	Validation cost measured and reported as a ratio to baseline pipeline duration	Paired pipeline runs with and without validation	[20]	Author-proposed

3.4. Assumptions and Architectural Boundaries

The design was subject to four assumptions. The estate is actually diverse and one catalogue will not be authoritative for every platform. Data ownership is distributed, controls need to be enforced without moving the data. Platform metadata is incomplete and often incorrect and the architecture records the provenance of metadata. And governance metadata is sensitive, which is why it applies controls to the control plane. The boundaries are also clear; the architecture defines logical components, interfaces and record types, but not the vendor products of these, not the query language, and not anything more than the roles that the controls need.

4. Proposed Scalable Data Governance Architecture

The architecture is broken down into four logical layers: a governed estate layer to store the data platforms; a data movement layer to ingest, orchestrate and transform data; a governance services plane to serve metadata, lineage, data quality, data policies and data stewardship services; and a consumption and oversight layer to generate analytical products, APIs, and reporting. The control plane contains the standards, identity and policy; data and processing continue to be distributed. Fig. 1 shows the components and their principal flows; the twelve components are enumerated in Section IV-A, and none has been implemented or evaluated.

4.1. Component Responsibilities

The estate layer includes on-premises databases and enterprise systems, cloud databases or object storage, data lakes and distributed processing platforms, and streaming channels; it is designed to provide catalogues, information schemas, transaction logs, orchestration histories and other harvestable interfaces it is not there to alter how data is stored. The arrival is registered in the ingestion and integrated layer which emits declared read-writer relationships. The orchestration layer provides the graph of the workflow and the run results. The metadata collection and standardisation services collect and normalize platform metadata and map the platform metadata to the canonical platform model, while the repository and catalogue maintain the canonical records and provide discovery. The lineage and provenance engine resolves and combines all declared, executed and inferred relationships into one graph. Versioned rules, profiling and evaluation are the responsibility of the quality service. The policy layer evaluates classification, access, masking and retention and propagates classifications along lineage. Stewardship services have their own terms, definition, ownership and exceptions. Monitoring, audit and reporting continue

with append-only evidence; coverages are exposed through interfaces; traversal and evaluation are exposed through interfaces; and governance metadata is governed by security controls.



Figure 1. Scalable Hybrid Enterprise Data Governance Architecture

There are two options that provide the majority of the coupling. The asset identity is a governance service, not a platform artefact: Every asset is given a canonical identity, and the platform-specific name is an attribute that may change without invalidating lineage, rules, classification associated with the asset. Lineage confidence is treated as a first-class attribute: every lineage edge contains an indication of the kind of relationship, whether it is declared or observed at execution or inferred, because a consumer needs to know the basis of a relationship, which is more important because the inference is a weaker signal than the declaration [12].

4.2. Lineage Capture and the Metadata Lifecycle

Lineage is recorded in three supporting mechanisms combined in one graph. Declarative capture parses definitions of pipelines and connectors and produces high quality edges for governed pipelines. Capture events for the orchestration and storage are consumed, and job, input, output and parameter relations are captured as they occur during execution. Similar to production systems, inferred capture analyses generated statements and inferred column-level relationships when no declaration exists, with a reduced confidence of candidate edges [24] [12]. Merging is by canonical identity and temporal ordering, a later observation is used to replace an earlier inference, not to completely wipe out the earlier inference. Where a relationship is expected but no channel supplies it, a gap object is written with a reason code, keeping incompleteness visible and countable rather than silently absent.

The metadata lifecycle is based on six stages: discovery lists assets; harvest collects technical and operational metadata on a schedule, recording the date of staleness bound to the asset; standardisation maps platform metadata to the canonical model, with a version of the mapping, plus a date of staleness bound to the mapping; curation adds business terms, definitions, and ownership to assets under stewardship; publication exposes assets to discovery, or policy-bound interfaces, and also records a date of staleness bound to the policy; archival holds versions, and audit evidence, under retention policy. Fig. 2 represents the lifecycle with collection points MP-1 to MP-6, the three types of records it generates and the transformation chain between the source asset and the analytical output.



Figure 2. End-To-End Data Lineage and Metadata Lifecycle

The three record types are deliberately distinct. The metadata record exists in the present, is mutable, and is authoritative. The lineage edge is a relationship with origin, confidence and observation time. Provenance information is append-only and is structured according to entity, activity and agent [10] and follows a domain-agnostic structure. The three are bound together into a single, canonical identity, making a governance assertion reconstructible, as illustrated in the appendix.

4.3. Data Quality Validation and Policy Enforcement

Quality controls are executed at four points instead of just one boundary. Schema, type and completeness rules are able to reject malformed arrivals before they propagate at ingestion. Transformation rules for joins, key integrity, and data derivation detect integrity violations during processing. At publication, quality rules assess whether the curated data asset meets the requirements of its intended use. In the context of consumption, rules and queries are used for testing fitness for use in the context of a specific consumer and the operational meaning of quality [19], [20]. Each rule is identified and versioned and each evaluation has the rule version, asset version, column version, population evaluated, threshold applied and outcome recorded; a compliance figure can be re-produced and challenged. Rules are grouped into rule sets per asset class, and a rule change follows the same approval path as a policy change, because a threshold moved silently is a governance change made without governance. Thresholds are configured values of the adopting organisation; this article proposes none.

Policy enforcement evaluates classification, access, masking, and retention requirements and records the applicable policy version and obligations. Classification is propagated along lineage so that derived assets inherit the stricter classification of their inputs, consistent with the approach described in [15]. When a data quality rule fails, the affected output is quarantined, the asset and evaluated population are recorded, and the responsible steward is notified. Revalidation follows remediation. Policy violations are handled separately because they require policy enforcement rather than data repair. Both paths generate audit records before the outcome is applied, preserving evidence if the action fails. Fig. 3 illustrates the workflow and its two paths.

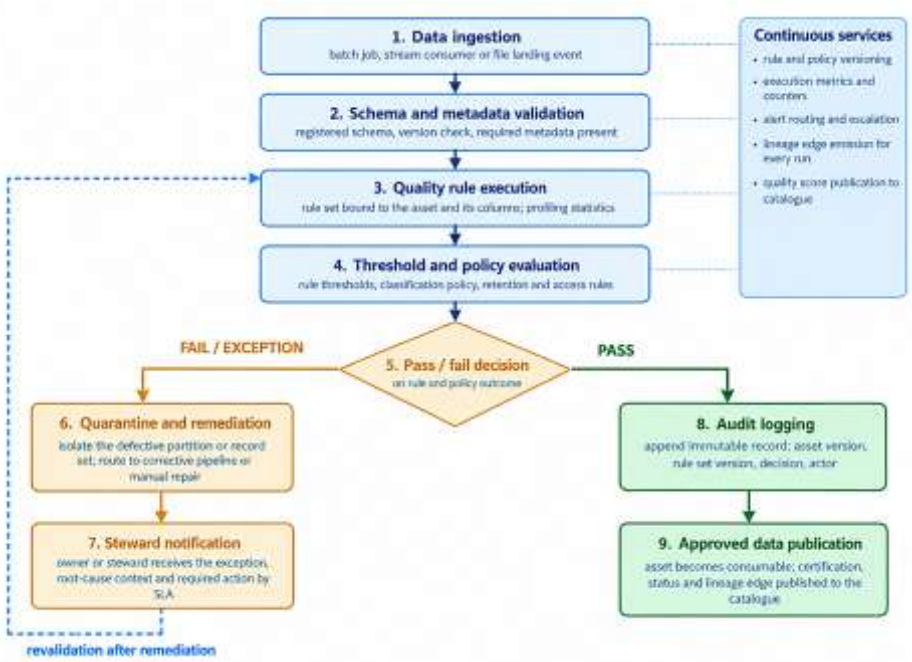


Figure 3. Integrated Data Quality Control and Governance Enforcement Workflow

4.4. Behaviour at Architectural Boundaries

Six stress cases are listed, as these are the stress cases in which governance architectures fail in practice. For schema evolution, the canonical record is set to a version, the previous version is stored, the column-level lineage is re-evaluated, and edges that have lost one (or both) of their endpoints are set to unverified instead of being deleted. For metadata-synchronisation lag, harvest returns the staleness bound and consumers get the observation time instead of silent staleness. For failed pipelines, run outcomes are marked as outputs that are incomplete, and quality evaluations are suppressed instead of failed, and lineage records an interval that is incomplete. If a gap is caused by a missing lineage, then the gap object will be raised with a reason code and a responsible component and gap counts will be reported as a coverage input instead of an unmeasured residual. When conflicting glossary definitions are detected, both definitions are retained, and the responsible owners are recorded so that the conflict can be resolved without losing evidence. This is because the definition is erased silently which removes the evidence required to resolve the definition. If there is a quality violation, the quarantine path is followed, and the violation is linked to the version of the rule that identified the violation.

4.5. Interfaces and Illustrative Data Model

Interfaces are defined by each component instead of being a single, monolithic API: asset and identity interfaces, platform connector harvest interfaces, lineage interfaces for edge reads and traversals, rule and evaluation interfaces for quality service, decision interfaces for policy service, and audit interfaces for evidence retrieval. Records of illustrative asset, lineage edge, gap object, provenance record, quality evaluation and policy decisions are built to demonstrate the model and are not from any production system.

5. Research Methodology and Evaluation Design

5.1. Review Method, Sources and Design Principles

This study is an evidence based architectural analysis. Sources were identified through a specific search of scholarly indexes and official documents in four categories governance/metadata frameworks, provenance/lineage research, quality standards, and quality assessment methods, and documented platform implementations. Each source was recorded with its bibliographic details, access level and the finding attributed to it, and no finding is attributed to a source whose text was not inspected. The review informed four design principles: separation of governance control from data custody, explicit representation of lineage incompleteness, identity and versioning of governance assertions, and clear differentiation between harvested and asserted evidence.

5.2. Evaluation Criteria and Measurement Definitions

Ten metrics are defined, each with a numerator, denominator, unit, measurement conditions and known limitations; all are proposed rather than measured.

$$LC = |R_{captured} \cap R_{expected}| / |R_{expected}| \times 100\% \quad (1)$$

Eq. (1) measures lineage completeness against an independently enumerated expectation set, so the denominator requires a reference enumeration; it measures agreement with that enumeration, not absolute completeness.

$$MC = |A_{with_required_metadata}| / |A_{in_scope}| \times 100\% \quad (2)$$

Eq. (2) measures metadata coverage per asset class; it is a presence measure, so a present but incorrect value scores the same as a correct one.

$$MCons = |M_{matching}| / |M_{compared}| \times 100\% \quad (3)$$

Eq. (3) measures agreement between repository and platform-native metadata over mapped attributes per platform pair and harvest cycle; a systemic harvesting defect can produce high reported consistency.

$$QRC = |N_{satisfying}| / |N_{validly_evaluated}| \times 100\% \quad (4)$$

Eq. (4) measures rule compliance per rule version and asset, excluding records that could not be evaluated, which must be disclosed separately because a weak rule yields high compliance.

$$PEC = |A_{decision_evaluated}| / |A_{governed}| \times 100\% \quad (5)$$

Eq. (5) measures policy enforcement coverage per policy version; it establishes that a decision was evaluated, not that the decision was correct.

$$LRL(d, n, e) = t_{response}(d, n, e) \quad (6)$$

$$MSL = t_{repository_reflected} - t_{source_change} \quad (7)$$

$$QVO = (t_{with_validation} - t_{baseline}) / t_{baseline} \times 100\% \quad (8)$$

$$GPT = N_{events_processed} / T_{observation} \quad (9)$$

$$FDR = (t_{alert} - t_{fault}) + (t_{restored} - t_{alert}) \quad (10)$$

Eqs. (6) to (10) are distribution and duration measures: traversal latency at stated depth, node and edge counts, reported as median and 95th percentile over at least three repetitions; synchronisation delay per channel, measurable only where source timestamps exist; validation overhead from paired runs with identical inputs; throughput at stated concurrency and event mix; and detection plus recovery time per injected fault class, with the two terms reported separately. For all ten metrics, future

evaluations should report the software versions, configuration, observation period, measurement conditions, and relevant limitations. No metric values were measured in this study. Acceptance thresholds must be defined by the adopting organisation before evaluation and should not be interpreted as universal standards.

5.3. Reproducibility, Validity and Evidence Extraction

Without the asset class and attribute set, a metadata coverage is meaningless and without the reference enumeration, a lineage completeness figure is meaningless. Validity is achieved through separation: values from previous systems are reported in a column or section; values from comparative reading are reported in a different column or section; proposed metrics are reported in a different column or section; proposed architecture are reported in a different column or section; there is no single column or section that compares different studies on different measures under different conditions. The following protocol is proposed for future evaluation; it was not executed in this study. Fig. 4 is a mapping of the twelve requirements to capability domains and the status using only the labels Proposed / Untested and Reported.



Figure 4. Governance Coverage and Requirement Traceability Matrix

6. Evidence-Based Architectural Analysis and Proposed Evaluation Protocol

6.1. Evidence-Based Governance Challenges

The evidence reviewed suggests that there are four findings regarding the conditions that the architecture must endure. Fragmentation is scale-invariant: even a catalogue that indexes over 26 billion datasets requires inference, and even user-provided descriptions of uneven richness more harvesting don't necessarily yield shared understanding [12]. Explicitness is load-bearing: When a metadata store is separate from the data, the platform needs to actively maintain consistency between the two, making it an engineered property, not emergent [11]. Incompleteness is permanent: lineage systems are designed to create partial traces and to trade off precision versus recall and assume that gaps in the model will be represented so as to be able to do so. But improvement is under-evidenced: thirteen quality methodologies have been compared, and gaps in strategy and a lack of evidence on the economics of quality work have been identified; thus, an expected benefit cannot be assumed when proposing a control [20]. The values that underlie these findings are listed in Table III in conjunction with their location, conditions and limitations.

Table 3. Source-Reported Findings and Quantitative Evidence

Reference	Location in Source	Research Context	Data or Platform	Reported Metric	Reported Value	Conditions As Reported	Interpretation in this Article	Evidence Limitation
[12]	Section 2.1 and Table 1 (read in full text)	Enterprise catalogue at internet scale	Internal structured datasets, >20 formats	Catalogued datasets	Over 26 billion	Counted over datasets readable by all engineers only	Metadata fragmentation persists at extreme scale	Internal access-restricted datasets excluded from the count
[12]	Table 1 (read in full text)	Catalogue ingestion rate	Storage path events	Paths added or deleted per	Over 1.6 billion	Steady-state operation	Harvesting capacity is a first-order	No breakdown by system or

				day		of the crawler	scalability constraint	peak-to-mean ratio given
[11]	Section 1 and Section 5 (read in full text)	Warehouse over a distributed file system	Production internal warehouse	Data volume, tables, users, queries	>700 TB usable; several thousand tables; >100 users; >5000 queries/day	As reported at the time of writing	An explicit metastore was required to make large-scale analysis usable	Snapshot figures; no growth or latency series
[14]	Section 1 (read in full text)	Enterprise curation of heterogeneous sources	Three real-world curation problems	Curation cost reduction	About 90% relative to deployed production software	Comparison against the software in production at the study sites	Automated curation can substitute for manual effort at scale	Cost basis not decomposed; no confidence interval reported
[14]	Section 5 (read in full text)	Attribute identification	Web aggregator sources	Correct attribute mapping after training	90%, trained on fewer than 50 manually labelled sources	Modest manual labelling effort assumed available	Automated metadata mapping is viable but needs labelled seeds	Single case; labelling cost not quantified in the accessible text
[14]	Figure 2 (read in full text)	Entity consolidation quality	Web aggregator data	Precision and recall	97% and 4% before tuning; 100% and 98.9% after (as shown in the figure)	Two operating points plotted in the figure	Quality improvements are threshold- and tuning-dependent	Operating-point selection criteria not restated in the accessible text
[17]	Figure 4 (read in full text)	Metadata operations on partitioned tables	Cloud object store table with up to 1 million partitions	Time to query table metadata	108 s at 1M partitions; 17 s with the log cached on SSDs; comparable systems exceeded one hour at 100k partitions	Benchmark configuration reported with the figure	Metadata-operation cost dominates at high partition counts and is fixable	Published 2020, outside the 2019 review period; used for retrospective context
[17]	Figure 5 (read in full text)	Data skipping through layout	500 TB network traffic dataset	Fraction of data skipped	93%	Multi-attribute queries with Z-ordering on four fields	Storage-layer metadata enables substantial pruning at query time	Workload-specific; not a general skipping rate
[17]	Figure 6 (read in full text)	Analytical query performance	TPC-DS, three runs per configuration	Power test duration	0.93 h for the Delta configuration versus 0.99 h, 1.44 h and	Average duration across three runs per configuration	Transaction-log metadata does not cost analytical performance	Vendor-run benchmark; hardware and tuning not generalisabl

					3.76 h for the compared configurations			e
[15]	Version 1.2.0 feature documentation (read directly)	Open metadata and governance platform	Hadoop ecosystem with external integration	Classification propagation along lineage	Stated as a feature: classifications propagate automatically along lineage	Documented capability of release 1.2.0, issued June 2019	Policy obligations can travel with the data rather than per platform	Documentation asserts capability; no measured propagation or enforcement figures
[16]	Project documentation (read directly)	Lineage metadata service	Jobs, datasets and runs across a data ecosystem	Metadata entities and interfaces	Run, job and dataset entities; HTTP API, UI and lineage graph; OpenLineage event format	Documented design of the service	A generic run-job-dataset model is an adequate substrate for enterprise lineage	Documentation is later than 2019; no performance or coverage measurement provided
[24]	Abstract only (not readable in full)	Lineage tracking for big data systems	Distributed processing jobs	Not reported in the accessible record	Not reported	Only the abstract-level description of the framework was accessible	The design goal of unifying database and workflow provenance is corroborated	No figures, pages or performance values verified; nothing is quoted from this source
[9]	Abstract only (full text not readable)	Systematic review of provenance systems	Multiple provenance systems compared	Not reported in the accessible record	Not reported	Only bibliographic and abstract-level information was accessible	Heterogeneity of provenance models limits interoperability	Comparative detail, tables and figures could not be inspected
[20]	Sections 2–6 and Table III (read in full text)	Comparative survey of quality methodologies	13 assessment and improvement methodologies	Methodologies compared	13, compared across six perspectives	Methodologies published between the 1990s and 2009	Quality practice is methodologically uneven and rarely empirical on cost	Survey of methods, not of outcomes; no measured improvement reported
[21]	Published scope statement (standard not purchased)	Data quality standardisation	Structured data in computer systems	Quality characteristics	Fifteen characteristics in two viewpoints (inherent and system-dependent)	Normative model as stated in the published scope	A shared quality vocabulary exists and is stakeholder-relative	Definitions and annexes were not accessible; only the scope was verified
[22]	Published scope statement (standard not purchased)	Data quality management standardisation	Organisations and processes	Process model	Processes required for data quality management, used to	Normative scope as published	Quality management is a process discipline, not a validation	The process reference model itself was not readable

					assess capability or maturity		step	
[23]	Principles section (read directly)	Decentralised data architecture	Enterprise analytical estate	Architectural principles	Four principles, including federated computational governance	Practitioner article, May 2019	Global standards with local ownership is a viable governance configuration	No implementation, adoption or measurement evidence

6.2. Comparison of Existing Governance and Metadata Approaches

Table I makes a comparison of the reviewed approaches, objective, framework approach, metadata approach, lineage capability, quality controls, reported findings and limitations and reveals four patterns instead of one trend. Governance research provides the decision-rights and accountability layer and no technical architecture [1, 2, 3]. Provenance and lineage research provides strong models such as a standard interchange model and formal differences between why, how and where provenance, but does not go so far as to operational hybrid deployments; at the same time, there are differences between the systems so that records cannot be ported from one system to the other [5, 6, 7, 8, 9, 10]. Quality work supplies standardized vocabulary and assessment method and recognizes that improvement and its economics is the weaker half [19, 20, 21, 22, 25, 18]. Most of the machinery needed to run these systems is provided by platform systems, each of which addresses a single platform family, including explicit metastores, extensible type systems, run-level lineage models, transaction-log provenance and large-scale catalogues, and all of which are local to that platform family [11, 15, 16, 17, and 12]. The architecture that fills that gap is the seam between those groups: identity, interfaces and record types that allow different operating systems to have the same set of definitions, the same lineage graph, and the same policy model.

6.3. Architecture-To-Requirement Traceability

Table IV maps each of the twelve governance requirements to the control that addresses it, the respective component of the control, the quality validation that is to be applied, the audit evidence that results from the quality validation, and the evaluation metric that would measure it and its verification status. Each row is marked Proposed / Untested as the table is to document design intent and planned evidence, not actual capability. The instrument of the architecture is the matrix: requirement mapping to no control is a governance gap and control mapping to no metric cannot be made to work.

Table 4. Architecture-To-Governance Control Traceability Matrix

Governance Requirement	Metadata or Lineage Control	Responsible Component	Quality Validation	Audit Evidence	Evaluation Metric
R1 Lineage across heterogeneous sources	Declarative parsing of connectors and ingestion definitions; execution events	Ingestion services; lineage engine	Structural validation of captured relationships	Edge origin and confidence per relationship	Eq. (1) lineage completeness
R2 End-to-end lineage across transformations	Statement-level inference plus run events merged into one graph	Lineage and provenance engine; orchestration	Gap detection and confidence marking	Gap objects with reason codes	Eq. (1); Eq. (6) traversal latency
R3 Technical and business metadata standardisation	Canonical metadata model with recorded platform mapping	Metadata standardisation; repository	Required-attribute checks per asset class	Mapping version and harvest record	Eq. (2) metadata coverage
R4 Metadata synchronisation and freshness	Scheduled harvest, staleness bounds, reconciliation job	Metadata collection; repository	Divergence detection and exception raising	Divergence events with both versions	Eq. (3); Eq. (7) synchronisation latency
R5 Quality controls in ingestion and transformation	Versioned rules bound to asset and columns	Data quality service	Rule execution in-stream and in batch	Rule version and outcome per evaluation	Eq. (4) rule compliance; Eq. (8) overhead
R6 Quality controls at consumption boundaries	Fitness-for-use rules bound to consumption paths	Data quality service; consumption interfaces	Pre-release validation of extracts and queries	Rule version, decision and consumer identity	Eq. (4); Eq. (5) enforcement coverage
R7 Policy enforcement	Classification with propagation along	Policy enforcement;	Policy decision recorded with	Policy version and decision	Eq. (5) policy enforcement

(access, retention, masking)	lineage; obligation attached to decisions	lineage engine	obligations	record	coverage
R8 Auditability and evidence retention	Append-only audit records linked to asset, rule and policy versions	Monitoring and audit logging	Reconstruction test on sampled decisions	Immutable evidence store with retention policy	Evidence reconstruction success (protocol level)
R9 Scalability with assets, pipelines and users	Domain partitioning of metadata and traversal; asynchronous integration	Repository; lineage engine; integration layer	Load and fault injection	Capacity and partition records per test run	Eq. (6); Eq. (9) throughput; Eq. (10) detection and recovery
R10 Schema evolution and failed-pipeline handling	Schema versioning; gap marking; suppression of expected output versions	Repository; orchestration; quality service	Impact evaluation of affected rules and edges	Change events and unverified-status records	Eq. (1) under change; Eq. (10)
R11 Traceability of definitions and ownership	Glossary terms bound to assets; ownership recorded per domain	Glossary and stewardship services	Coverage and conflict register audits	Term-to-asset mappings and conflict entries	Eq. (2) applied to business metadata
R12 Observability of governance operations	Per-service health and coverage signals to a common surface	Monitoring and reporting; all services	Governance-of-governance reporting	Service health and coverage reports	Eq. (5); Eq. (9)

6.4. Data Lineage and Metadata Coverage Analysis

No lineage or metadata coverage value is reported here, as there was no estate, pipeline set or asset inventory available and any number would be an arbitrary guess. The structure of the measurement is what is supported by the evidence. The completeness of the lines needs to be compared with an expectation set of lines, not with what the engine has captured. The coverage needs to be specified by an asset class and an attribute set, otherwise it's just one number that doesn't reveal the distinction between a technical field that has been populated and a business definition that has. Consistency needs to be specified over a period of time, for a repo value and a platform value can change without notice.

6.5. Data Quality Control and Policy Enforcement Analysis

The evidence indicates that multiple points for quality controls may be better placed than a single point, as the quality evaluation methods reviewed in this paper view quality as a process that is applied where data is generated and consumed [20, 25] and as quality is defined by the consumer, consumption is a valid point for evaluation [19]. And it supports propagating classification along lineage, not re-evaluating it per platform, as the latter leaves derived assets without a classification until someone re-classifies them. [15] No evidence exists to support any claims that these controls result in reduced error rates, latency or cost; the comparative survey explicitly mentions the thinness of evidence [20].

6.6. Proposed Evaluation Protocol

This subsection does not require an evaluation, but rather sets out the protocol that could be used to carry out one. Firstly, instrument a bounded hybrid estate of at least three types of platform with a declared asset inventory, and also report the asset inventory completed overall and per channel type, with gap counts separated. Second, harvest metadata for the inventory and report Eq. (2) per asset class and attribute set as well as harvest Eq. (3) after a deliberate change at a source. Thirdly, set up a rule set for each asset class, add known defects (schema breaks, key violations, late arrivals) and report Eq. (4) against the number of defects added, without giving the ground truth compliance percentages. Fourth, validate Eq. (5) for every policy version and for propagation of tests, make sure that a classified source attribute triggers the appropriate obligation on a derived asset that is at least three hops from the source.

Fifth measure the operational cost: Eq. (6) after one, three and five hops, Eq. (7) per channel, Eq. (8) for a pair of injected pipeline runs with same input, Eq. (9) at specified event mix at concurrency, Eq. (10) per injected fault class with separate detection and recovery. Sixth, once steps 1-5 are in place, do a controlled before/after evaluation against the other part of the estate where the same architecture is not enabled, to allow differences to be attributed to governance. Acceptance values are set by the adopting organization before measurement, and a reconstruction test on sampled historical decisions is the acceptance test for auditability.

7. Discussion

7.1. How the Architecture Addresses the Identified Gaps

The architecture is designed to respond to RQ1: to integrate three capture mechanisms and to depict both confidence and gaps, as completeness needs to be engineered. It resolves RQ2 by providing a canonical model and mapped platforms are recorded on the platform, which eliminates the need for one platform to impose its schema. It addresses RQ3 by associating versioned rules to versioned canonical assets and keeping the version of rules, assets, and columns in each evaluation. It is used to answer RQ4 and performs the same evaluation as quality by comparing it at the same points, while it propagates classification along lineage. It only responds to RQ5 in design: partitioning by domain and measuring coverage, latency and throughput separately is the mechanism; the answer remains unverified until the protocol in Section VI-F is run.

7.2. Trade-Offs

Centralised metadata vs federated metadata. Federation is easy to scale, but leads to a loss of consistency and duplicated harvesting, while centralisation means less complexity with identity, standardisation, and traversing multiple domains, and risk of being a single point of failure and bottle neck for contested definition. The architecture strikes a balance: global standards and identity, local ownership of definition and quality rules, which is the decentralised argument that standards should be global and accountability should remain local [23] and the contingency argument that no single design can fit every organisation [2]. Lineage: real-time vs. batch.

Real-time capture makes impact analysis possible in a short time window, but requires streaming instrumentation and is more expensive per event; batch capture is less expensive, but impact analysis is based on stale lineage, which is acceptable for reporting, but not for access control, so the staleness bound must be published. Automated or Curated metadata. Automation scales but is less confident about the inference, and the business-metadata sub-set that decisions are based on is already uneven at internet scale [12]; so only stewardship is needed for the business-metadata sub-set. Control versus autonomy. Each control point eliminates the opportunity for uncontrolled risks and introduces latency; the nonfunctional requirement limits overhead, and the consumption-point rule set is limited to the consumer and not applied across all consumers.

7.3. Implications for Stakeholders

For enterprise data architects, governance should be implemented as a control plane, not a platform, with explicit interfaces to survive platform changes. The governance officers have the versioned rule and policy register with approvals as the instrument of operation, because Table IV provides an audit-oriented view of proposed control coverage; it does not demonstrate that the controls have been implemented or validated. Data stewards, meanwhile, are going from describing large quantities of data to addressing the conflicts and gaps that automation reveals, a more manageable workload. Propagated classification fills the gap in the derived-asset exposure pathway left by per-platform labelling for security teams.

7.4. Conditions for Usefulness and Evidentiary Status

The architecture will be useful if there are three or more platforms involved, if regulatory or contractual requirements demand reconstructible evidence of the way the output is made, and if it is necessary to have a central team to control the metadata, but that team is not the owner of the platform. It has concrete dependencies: platforms should offer accessible harvestable metadata and execution history, asset identity has to be determined first, so that lineage, rules and classifications can then be bound, and stewardship must be in place, as the architecture sends exceptions to someone, not to the computer. The status of evidence should be clearly expressed.

The literature supports the design principles and mechanisms: explicit metadata-data consistency, classification propagation along lineage, the standard structure of provenance records, and the multi-strategy nature of quality work, the contingency of governance design and the unevenness of supplied metadata. What this article suggests, and hasn't proved, is that these mechanisms can be integrated into a single architecture with acceptable coverage, latency and overhead at the enterprise scale. A hypothesis for future evaluation is that partitioning governance metadata by domain while maintaining global asset identity and shared standards can limit lineage traversal latency and metadata harvesting contention as the number of assets increases. This hypothesis can be tested using Eqs. (1), (6), and (9), but it has not been established by the sources reviewed. The figure 5 represents a closed loop in which violations detected should become rule, threshold, or policy changes, and it is proposed, but no cycle time, coverage figure, or improvement rate has been measured.



Figure 5. Closed-Loop Data Governance Monitoring and Improvement Cycle

8. Threats to Validity and Limitations

- Literature selection bias. Sources were prioritised by verifiability and relevance, favouring well-indexed venues and ensuring English-language work, and may under-represent vendor and practitioner evidence, but with some mitigation provided by platform documentation.
- Incomplete source access. Many of the sources were only available at abstract or scope level, and one source was accessible PDF that was not machine-readable; no finding is attributed to these sources beyond their verifiable bibliographic record, limiting the scope of comparison in Table I.
- Heterogeneity of implementations and semantics. Platforms differ in metadata models, lineage semantics, granularity and asset identity, so the mappings proposed here are design proposals rather than verified equivalences, and a mechanism that works in one platform family may not transfer to another.
- Incomplete source metadata, schema evolution and dynamic pipelines. The architecture assumes platform metadata is sometimes wrong and lineage sometimes missing; inference and gap records manage this but do not eliminate it, and dynamic statements remain the hardest case.
- Scalability, overhead, security and access restrictions. The control plane does not use design reasoning as metrics: partitioning, latency and overhead arguments are not metrics, but governance metadata is sensitive, and access-control constraints are imposed on it that restrict who can verify it.
- Lack of empirical assessment and of real enterprise data. No implementation, workload, fault injection, or case study was available. All metrics in Section V-B are therefore proposed but not measured. The effectiveness, cost, and scalability of the proposed architecture remain unvalidated hypotheses. The evaluation protocol is reproducible, but no experimental results are available.

9. Conclusion and Future Work

The distributed nature of enterprise data platforms, each of which may have different data schema, data stores, lineage mechanisms and access semantics, have a cost and the evidence reviewed shows these costs: Fragmented metadata at internet scale, consistency of metadata and data being an explicit responsibility, inconsistent lineage being only as complete as it is engineered, and the cost of improving data quality not being well established. This article proposes an architecture that couples a canonical asset identity to metadata records, a set of confidence-bearing lineage edges, a set of versioned quality rules and propagated policy classifications, organised into four levels and twelve components, with a set of defined behaviour under six stress cases, expressed in a set of ten reproducible metric definitions and a requirement-to-control traceability matrix. It adds a design which is specified and falsifiable, and a rationale that is separated from the evidence; it does not add evidence of a performance result: no implementation was constructed, no experiment was conducted, no measurement was taken, and quantitative values are reported by cited sources under their own conditions. What can be stated with evidence is narrower and more useful: standardised provenance structure, explicitly maintained metadata-data consistency, classification propagation along lineage and versioned quality rules are each supported by documented practice and research, and their combination into one requirement-mapped architecture for hybrid platforms is the gap this article addresses.

Future work should proceed in four directions: implementation and workload-based scalability testing of the repository, lineage engine and policy services at reported asset, pipeline and event scales, measuring Eqs. (1), (6) and (9) under stated

configurations; metadata interoperability benchmarks comparing canonical mappings across platform families and quantifying mapping loss; automated lineage validation covering dynamic statements with a measured precision-recall trade-off for inferred edges; quality-rule orchestration experiments with injected defects to test attribution and overhead, using Eq. (4) against known ground truth;

References

[1] V. Khatri and C. V. Brown, "Designing data governance," Commun. ACM, vol. 53, no. 1, pp. 148–152, Jan. 2010, doi: 10.1145/1629175.1629210.

[2] K. Weber, B. Otto, and H. Österle, "One size does not fit all A contingency approach to data governance," ACM J. Data Inf. Qual., vol. 1, no. 1, art. 4, Jun. 2009, doi: 10.1145/1515693.1515696.

[3] R. Abraham, J. Schneider, and J. vom Brocke, "Data governance: A conceptual framework, structured review, and research agenda," Int. J. Inf. Manage., vol. 49, pp. 424–438, Jul. 2019, doi: 10.1016/j.ijinfomgt.2019.07.008.

[4] DAMA International, DAMA-DMBOK: Data Management Body of Knowledge, 2nd ed. Basking Ridge, NJ, USA: Technics Publications, 2017, ISBN 978-1-63462-234-9.

[5] Y. Cui, J. Widom, and J. L. Wiener, "Tracing the lineage of view data in a warehousing environment," ACM Trans. Database Syst., vol. 25, no. 2, pp. 179–227, Jun. 2000, doi: 10.1145/357775.357777.

[6] P. Buneman, S. Khanna, and W.-C. Tan, "Why and where: A characterization of data provenance," in Database Theory ICDT 2001, LNCS 1973. Berlin, Germany: Springer, 2001, pp. 316–330, doi: 10.1007/3-540-44503-X_20.

[7] J. Cheney, L. Chiticariu, and W.-C. Tan, "Provenance in databases: Why, how, and where," Found. Trends Databases, vol. 1, no. 4, pp. 379–474, 2009, doi: 10.1561/1900000006.

[8] M. Herschel, R. Diestelkämper, and H. Ben Lahmar, "A survey on provenance: What for? What form? What from?" VLDB J., vol. 26, no. 6, pp. 881–906, Dec. 2017, doi: 10.1007/s00778-017-0486-1.

[9] B. Pérez, J. Rubio, and C. Sáenz-Adán, "A systematic review of provenance systems," Knowl. Inf. Syst., vol. 57, no. 3, pp. 495–543, Dec. 2018, doi: 10.1007/s10115-018-1164-3.

[10] L. Moreau and P. Missier, Eds., "PROV-DM: The PROV data model," W3C Recommendation, Apr. 30, 2013. [Online]. Available: https://www.w3.org/TR/prov-dm/

[11] Thusoo, J. Sen Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy, "Hive—A warehousing solution over a map-reduce framework," Proc. VLDB Endowment, vol. 2, no. 2, pp. 1626–1629, Aug. 2009, doi: 10.14778/1687553.1687609.

[12] Halevy, F. Korn, N. F. Noy, C. Olston, N. Polyzotis, S. Roy, and S. E. Whang, "Goods: Organizing Google's datasets," in Proc. ACM SIGMOD Int. Conf. Management of Data, San Francisco, CA, USA, 2016, pp. 795–806, doi: 10.1145/2882903.2903730.

[13] J. M. Hellerstein et al., "Ground: A data context service," in Proc. 8th Biennial Conf. Innovative Data Systems Research (CIDR), Chaminade, CA, USA, Jan. 2017.

[14] M. Stonebraker, G. Beskales, D. Bruckner, M. Cherniack, I. F. Ilyas, S. Zdonik, A. Pagan, and S. Xu, "Data curation at scale: The Data Tamer system," in Proc. 6th Biennial Conf. Innovative Data Systems Research (CIDR), Asilomar, CA, USA, Jan. 2013.

[15] Apache Software Foundation, "Apache Atlas 1.2.0 Data governance and metadata framework for Hadoop," documentation, released Jun. 6, 2019. [Online]. Available: https://atlas.apache.org/1.2.0/

[16] Marquez Project, "Marquez: An open source metadata service for the collection, aggregation, and visualization of a data ecosystem's metadata," GitHub repository; OpenLineage, "An open standard for metadata and lineage collection." [Online]. Available: https://github.com/MarquezProject/marquez and https://github.com/OpenLineage/openlineage

[17] M. Armbrust et al., "Delta Lake: High-performance ACID table storage over cloud object stores," Proc. VLDB Endowment, vol. 13, no. 12, pp. 3411–3424, Aug. 2020, doi: 10.14778/3415478.3415560.

[18] Apache Software Foundation, "Apache Griffin: Big data quality solution for batch and streaming." [Online]. Available: https://griffin.apache.org/

[19] R. Y. Wang and D. M. Strong, "Beyond accuracy: What data quality means to data consumers," J. Manage. Inf. Syst., vol. 12, no. 4, pp. 5–33, 1996, doi: 10.1080/07421222.1996.11518099.

[20] Batini, C. Cappiello, C. Francalanci, and A. Maurino, "Methodologies for data quality assessment and improvement," ACM Comput. Surv., vol. 41, no. 3, art. 16, Jul. 2009, doi: 10.1145/1541880.1541883.

[21] ISO/IEC 25012:2008, Software Engineering Software Product Quality Requirements and Evaluation (SQuaRE) Data Quality Model. Geneva, Switzerland: ISO/IEC, Dec. 2008. [Online]. Available: https://www.iso.org/standard/35736.html

[22] ISO 8000-61:2016, Data Quality Part 61: Data Quality Management: Process Reference Model. Geneva, Switzerland: ISO, Nov. 2016. [Online]. Available: https://www.iso.org/standard/63086.html

[23] Z. Dehghani, "How to move beyond a monolithic data lake to a distributed data mesh," martinfowler.com, May 20, 2019. [Online]. Available: https://martinfowler.com/articles/data-monolith-to-mesh.html

[24] M. Tang, S. Shao et al., "SAC: A system for big data lineage tracking," in Proc. IEEE 35th Int. Conf. Data Engineering (ICDE), 2019, pp. 1964–1967, doi: 10.1109/ICDE.2019.00215.

[25] L. L. Pipino, Y. W. Lee, and R. Y. Wang, "Data quality assessment," Commun. ACM, vol. 45, no. 4, pp. 211–218, Apr. 2002, doi: 10.1145/505248.506010.

Appendix: Illustrative Data Model Records

The records below are constructed to illustrate the data model referenced in Section IV-E. They are not extracted from any production system and contain no real asset names, no real owner identifiers and no measured values.

Z	Illustrative serialisation (constructed for illustration; not production data)
Canonical asset record	{ "asset_id": "urn:gov:asset:0001", "qualified_name": "warehouse.sales.fact_order", "platform_refs": [{"platform": "onprem-warehouse", "object": "db.schema.fact_order"}, {"platform": "cloud-object-store", "object": "s3://curated/sales/order/v3"}], "asset_class": "curated_table", "schema_version": "v3", "owner_domain": "sales-operations", "steward": "steward@sales.example", "classification": ["internal", "pii:indirect"], "business_terms": ["Order", "Net revenue"], "quality_status": {"rule_set": "dq-order-v4", "last_evaluated": "T-1", "result": "pass"}, "metadata_origin": {"structural": "harvested", "business": "curated"} }
Lineage edge	{ "edge_id": "urn:gov:edge:7f21", "from_asset": "urn:gov:asset:0001", "to_asset": "urn:gov:asset:0142", "granularity": "column", "attribute_map": [{"from": "net_amount", "to": "revenue_net"}], "origin": "execution_event", "confidence": 1.0, "gap": false, "produced_by_run": "urn:gov:run:9a03", "observed_at": "T" }
Gap object (unresolved relationship)	{ "gap_id": "urn:gov:gap:1188", "asset": "urn:gov:asset:0311", "expected_input": "unresolved", "reason": "dynamic_sql_not_parsed", "confidence": 0.0, "routed_to": "platform-engineering", "status": "open" }
Provenance record (entity–activity–agent)	{ "entity": {"id": "urn:gov:asset:0142", "version": "v7"}, "activity": {"id": "urn:gov:run:9a03", "job": "etl-sales-order-daily", "started": "T-2", "ended": "T-1"}, "agent": {"id": "svc-airflow", "type": "software_agent"}, "relations": ["wasGeneratedBy", "used", "wasAssociatedWith"], "input_entities": ["urn:gov:asset:0001@v3"], "parameters": {"partition": "T-2", "code_hash": "sha256:..."} }
Quality evaluation record	{ "asset": "urn:gov:asset:0001", "rule_set_version": "dq-order-v4", "point": "pre_publication", "evaluated_records": 4123000, "validly_evaluated": 4101500, "not_evaluable": 21500, "satisfying": 4088440, "decision": "fail",

	"action": "quarantine_partition", "window": "T-1 daily" }

Policy decision record	{
	"asset": "urn:gov:asset:0142",
	"policy_version": "retention-and-masking-v11",
	"point": "consumption",
	"decision": "allow_with_obligation",
	"obligations": ["mask:net_amount", "retention:7y"],
	"subject": "role:analyst",
	"audit_ref": "urn:gov:audit:55120"
	}